Printed: 11-11-2004 ODOCP 02019563

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 465 011 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 91305122.3

(51) Int. Cl.5: H04N 1/32, H04N 1/42

(22) Date of filing: 06.06.91

(30) Priority: 26.06.90 US 543998

(43) Date of publication of application:
08.01.92 Bulletin 92/02

(84) Designated Contracting States:
DE FR GB

(71) Applicant: Hewlett-Packard Company
Mail Stop 20 B-O, 3000 Hanover Street
Palo Alto, California 94304(US)

(72) Inventor: Burgess, Ken L.
707 Peterson Street
Fort Collins, CO 80524(US)
Inventor: Marvin, John S.
4216 Picadilly Drive
Fort Collins, CO 80526(US)

(74) Representative: Colgan, Stephen James et al
CARPMAELS & RANSFORD 43 Bloomsbury Square
London WC1A 2RA(GB)

(54) Method of encoding an E-mail address in a fax message and routing the fax message to a destination on a network.

(57) A fax message transmitted by a facsimile transmitter includes bar coded headers in its first page. At least one of these headers contains the name of an addressee that is also a user on a network. A fax server receiving the incoming fax message inspects the first page of the incoming facsimile to locate the bar coded headers. If a TO: header is found it is used to determine the corresponding E-mail address, and the fax is automatically routed as E-mail on the network to the addressee. Any other headers, such as a FROM: or SUBJECT: header have their bar coded content converted to ASCII and attached as ASCII strings to the first page for easy inspection. An asymmetrical nature of the bar code used allows the fax server to determine which of a left-to-right or right-to-left scanning direction produces valid bar code. This in turn indicates whether the headers for the first page are right side up or upside down. By implication, this determines the orientation for the entire fax document. If the document is found to be upside down the fax server erects the document before mailing it to the addressee. The fax server or some other application running on a computer served by the network may be the addressee, and if the incoming fax is a request for information (whether by further bar code or check marks in predefined fields) the information may simply be sent by return fax to the sender, perhaps as part of the same phone call.

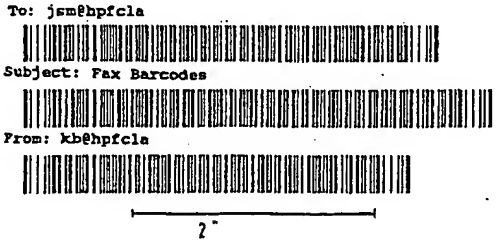

FIG 8A

EP 0 465 011 A2

1 06.11.2004

Printed: 11-11-2004 EPODOC 02019563

Background of the Invention

The transmission of messages and documents by facsimile transceivers (fax machines) is becoming an increasingly popular and widespread practice. As a further development, it is not necessary for an actual fax machine to exist at either the source or the destination of the message transaction. A so-called "fax modem" may take the place of the fax machine at either or both locations. A fax modem connects to a telephone line on one side and to a computer on the other. On the telephone side it can both send and receive audio signals in one of the CCITT formats that represent an image undergoing facsimile transmission. On the computer side it either sends or receives a stream of eight-bit bytes of compressed data representing an image. A suitable application program running on the computer either converts an incoming byte stream to a bit-mapped image for display on a monitor or other output device, or converts a displayed image to its corresponding outgoing byte stream. Such an application program (together with its associated computer) may be termed a "fax viewer."

In a large facility it is typical to find many fax modems connected on the phone line side to some number of phone lines, and connected on the computer side to a single computer running a program called (in association with the computer) a "fax server". In turn, the fax server is connected to a Local Area Network (LAN), wide area network, or other Electronic mail (E-mail) system. (Hereinafter, we shall simply use the term "network" as a synonym for any of the preceding computer-to-computer communication arrangements.) In such a facility there may be several hundred subscribers on the network, each with a fax viewer running on his computer and an assigned fax phone number, but perhaps only five or ten fax modems. Whereas a small business may well be obliged to use an actual subscriber loop per modem (which may be termed "traditional single party service"), a larger facility can arrange for the local phone company to interact with the facility as if it were an exchange in its own right. Then, some level of trunking is provided between the central office of the local phone company and a private branch exchange at the large facility. A well known conventional arrangement allows the allocation of the fax modems to whichever ones of selected (for fax service) phone lines are in use. This arrangement may be termed "direct inward dialing."

With either traditional single party service or direct inward dialing there is a necessary correspondence between the phone number used to dial the destination within the facility and the E-mail address of the intended recipient. If someone wants to send Charlie (who is a useful but wholly fictitious person) a fax, then they must dial Charlie's fax number. To correctly route the received fax the fax server must determine to which phone line the modem receiving Charlie's incoming fax is connected. That information indexes a table of E-mail addresses; Charlie's fax phone number points to Charlie's E-mail address.

While single party service and direct inward dialing work, they are not without a few warts. First, there is the matter of the phone line per user for single party service. To add a user it is necessary to add a phone line, which involves the phone company and added expense. Even with direct inward dialing a proliferation of fax phone numbers could require a change to the level of trunking to maintain adequate service. It would be cheaper and just as useful if a way could be found to use a suitable number of phone lines all representing the same phone number, and answer or select them "in rotation." Second, unless there is a modem per phone line (expensive!), there is the hardware needed to keep track of which phone line is being served by which fax modem. The system would be less expensive if that hardware could be eliminated. Third, there is system administration associated with adding or deleting a user. The table of correspondence between fax phone numbers and E-mail addresses must be kept up to date, or incoming faxes will not be correctly routed.

And while on the subject of warts, it should be noted that a conventional fax server has no way of determining if the received fax message will appear upside down on Charlie's fax viewer. It is usually left to the recipient to turn upside down messages right side up.

The underlying reason for all these particular warts on the prior art is that the fax server has no way to read the incoming fax message and intelligently base its actions on the contents of the message.

Summary of the Invention

We have a devised a way of making the fax server responsive to certain contents that may be placed into the fax message. Our fax server expects a fax message to include as its first page a header page, which in turn may contain the information to which the fax server is responsive. The incoming fax includes on its header page bar coded information that is detected and read by the fax server. This information includes, but is not limited to, a TO: line and a FROM: line, as well as a SUBJECT: line. These lines are logical lines, and a mechanism has been provided to allow a logical line of bar code to be composed of

2 06-11-2004

Printed: 11-11-2004 ODOCP 02019563

several physical lines of bar code. Of these logical lines, it should be noted that only the TO: line (or one having a corresponding logical function) is truly necessary for the automatic routing of incoming fax's. A TO: line is used to specify the E-mail address to which the fax message is to be routed.

Because the actual E-mail destination address can now be put into the fax (via a TO: line in the header) and then recovered by the fax server, no table of correspondence between phone numbers and E-mail destination address is required. In fact, the need for direct inward dialing for fax service is itself eliminated, since the fax server can now route an incoming fax irrespective of which phone line the fax was received on. The facility can now distribute a single fax number for all incoming fax traffic, and respond to that number with an appropriate number of phone lines answered in rotation. Lines can be added or deleted to reflect the volume of traffic without disturbing the arrangements through which persons communicate by fax. That is, Charlie doesn't need his own special fax phone number to receive fax messages on his E-mail terminal. Obviously, gone also is the mechanism that informs the fax server of which phone line the fax modem answered.

Also, no special system administration is needed, over and above the normal amount associated with Charlie's having an E-mail address in the first place. That is, the fax server need not maintain knowledge of particular E-mail addresses at all, and if desired, it can be left up to the E-mail system to decide what to do with a message having a defective address.

It may be argued that not every sender will be equipped with the proper bar code generation mechanism, and that too many messages will end up in the unknown or general delivery bin. Suppose Charlie gets a voice phone call from someone who wants to send him a fax, but who does not have any means to create the needed bar code to correctly address Charlie on E-mail. Then Charlie just faxes (!) a suitable cover sheet to the other party, who then makes as many photo-copies as desired and uses one to send his message to Charlie. The bar code reader in our fax server is quite tolerant of size changes in the bar code, and enlargements or reductions in size and skew arising from repeated photo-copying cause no harm.

Another aspect of the invention is that the particular bar code preferred for use in this method is not symmetrical for a certain "START/STOP" cipher used to enclose each physical line of bar coded information. Thus it is possible for the fax server to inspect a scan line of bar code looking for a START/STOP cipher, and do it from both left-to-right and right-to-left. Owing to the asymmetry, only one direction of inspection will be successful. Since the START/STOP cipher is on the left side of the line when the page is right side up, which inspection is successful indicates whether or not the header page (and by implication, the entire document) is right side up or upside down. If it is upside down the fax server can turn it right side up before sending it to its E-mail destination.

Further aspects of the invention are as follows. For the convenience of the human reader at the destination, the various bar codes are supplemented by their ASCII string counterparts, so that the TO:, FROM: and SUBJECT: lines on the cover sheet can be read and understood by Charlie when he looks at the message.

Now, it may be the case that the addressee is not a human being at all, but a piece of software designed to respond to things sent to it by fax. There are at least two ways this can work. First, the software could be in a computer that is some destination on the network, just as any other addressee. The message that this software receives is itself bar code, check boxes in a predefined field, or a combination of both. It could also include font characters to be recognized by optical character recognition. Second, the software could be in the fax server itself. In this case the fax server does not route the message, but directly performs itself any action needed in response to the message. In this connection we have provided that our fax server detect a particular cipher in a designated line of the bar coding in the header page, say on the first line. This cipher indicates whether to route a fax to an E-mail address or treat the entire fax as message to the fax server itself requiring some particular response.

Such a fax server can be used to perform a variety of functions. Consider the lowly advertising "bingo" card, which is a post card included in, say, a magazine. The reader fills out the card and mails it. In due course some requested goods arrive or a requested service is rendered. With the aid of the fax server described herein, the reader could fill out his bingo card and fax it in. And, if the request was for information in printed form, then the fax server could simply fax it back, on the spot! In similar fashion, one could use the inventive system to accomplish order placement generally, and as an overall replacement for optical character recognition at the resolution of fax machines. With the aid of a program that generates from ASCII characters graphics images that are bar code, a sender of such faxes to a fax server can generate for arbitrary ASCII text the corresponding bar code for inclusion in the message.

Brief Description of the Drawings

3 06-11-2004

Printed 11-11-2004 EDOCP 02019563

Figure 1 is a simplified block diagram of a fax server/network installation for use with the method of the invention;
Figure 2 is a simplified flow chart of an idle loop in the fax server of Figure 1;
Figure 3 is a simplified flow chart of a process started by the flow chart of Figure 1 and used to determine that the modem of Figure 1 is receiving an incoming fax message;
Figure 4 is a very simplified flow chart of a process started by the flow chart of Figure 1 and used to actually receive an incoming fax message from the modem and leave the message in a file for further processing;
Figures 5A-B are a flow chart of the process ROUTE FAX that appears as a step in the flow chart of Figure 2;
Figures 6A-B are a flow chart of a procedure named BAR CODE that is used as a component operation in the flow chart of Figures 5A-B;
Figure 7 is an example of a line of bar code that is used to represent header information in accordance with certain aspects of the invention;
Figures 8A-D are examples of a sample header page having different sized bar code ciphers for header information to be used in accordance with certain aspects of the invention;
Figure 9 is an illustration of the ciphers of a particular bar code preferred for use with the invention; and
Figure 10 is an illustration of an information request "bingo card" incorporating aspects of the invention.

Description of a Preferred Embodiment

Refer now to Figure 1, wherein is shown a simplified block diagram of one system to which the invention is applicable. As shown in Figure 1, a number of subscriber's loops (1, 2, 3) originate from a local telephone company outside the user's facility. These subscriber's loops 1-3 are what are commonly thought of as "telephone lines," although it will be understood that the practice of the invention is not in any way limited to use with only metallic conductors such as those employed in traditional telephone systems. That is, the facility's telephone service can be carried over any suitable communications channels using any appropriate technology; e.g., microwaves or fiber optics. Furthermore, it will be understood that while the figure is compatible with the case where each of the subscriber's loops has its own phone number (direct inward dialing), it is in no way limited to just that situation. Figure 1 is equally representative of the situation where each one of the subscriber's loops 1-3 represents the same phone number. In fact, it is just this latter situation that is preferred, since it dispenses with the inconvenience of distributing more than one phone number for fax communication.

After the subscriber's loops 1-3 cross into the user's facility they each terminate at a respective fax capable modem 4-6. By the term "fax capable" we mean that the modems 4-6 can at a minimum send and receive fax images, but may be capable of other types of service as well. And although we have shown one modem per subscriber loop, other arrangements may be possible involving a switching arrangement (not shown). Such a switching arrangement would allow there to be fewer modems than subscriber loops. Each of the modems 4-6 is connected to a computer 7, which, in conjunction with the software that it runs, is called the fax server. The computer 7 serves a network 8 upon which resides a number of users 9-11. The users 9-11 all have E-mail addresses and terminal or workstation hardware and associated software suitable for receiving fax images; e.g. high resolution monitors, laser printers and fax viewer software.

It will, of course, be understood that Figure 1 is a generalization of a wide variety of possible arrangements for connecting a computer network to a telephone system for the sending and receiving of fax messages.

Turn now to Figure 2, which is a simplified flow chart of the main idle loop for the fax server 7 of Figure 1. Actually, in one real installation it is a flow chart for the highest level of abstraction associated with the fax server function executed on the computer 7; there may be other functions performed by the computer 7, such as managing the E-mail system on the network 8 or maintaining some bulletin board. Indeed, the function of "fax server" is accomplished by running an appropriate application program on the computer 7; any of these other functions would be accomplished by "simultaneously" running other corresponding application programs on the computer 7, too. To this end, the operating system for computer 7 is of the multi-tasking variety. The details of what amount to time sharing between these various other applications and the fax server software are handled by the operating system. With the foregoing in mind, it will be understood that Figure 2 is the main idle loop for the fax server function of the computer 7, and that there may well be other main idle loops of equal logical rank (or position in a hierarchy) in existence on the computer 7. That is to say, the flow chart of Figure 2 is not the supreme idle loop of the highest possible abstraction on the computer; that idle loop would be someplace in the operating system itself. Instead, the

4 06-11-2004

Printed: 11-11-2004 EPOCP 02019563

flow chart of Figure 2 is the highest level of abstraction for a given category of application program activity, which in this case is fax message management.

The flow chart of Figure 2 is an arrangement of ten steps 12-21; in accordance with convention, decision-making is represented by diamonds and the performance of tasks by rectangles. It will be understood that the flow chart of Figure 2 has been simplified along functional lines, and that an actual program carrying out the stated functionality would probably deserve a longer and more detailed flow chart, principally because of the details of implementation.

Step 12 determines if there is new incoming activity on one of the modems. Exactly how this may be accomplished is somewhat implementation dependent; an interrupt service routine, for example, could detect the sending of an ASCII string from a modem previously listed as dormant. Once step 12 has determined that there is new (i.e., so far unacknowledged) incoming activity on a modem, step 13 schedules a process named IDENTIFY whose purpose is to acknowledge the activity and find out what kind of traffic the modem is to handle. The process IDENTIFY is an independently executable program to be performed by the computer 7 under its multi-tasking operating system. The process IDENTIFY is discussed in connection with Figure 3. Upon its conclusion, IDENTIFY returns to step 12 of the flow chart of Figure 2.

Continuing with Figure 2, if step 12 determines that there is no unacknowledged activity at any of the modems, step 14 determines if a newly acknowledged activity is a fax message. The way this works is that there has been a previous instance where step 12 scheduled IDENTIFY, which in turn determined that the incoming activity was indeed a fax and created some interprocess indication (flag, message in a mailbox in memory, etc.). Step 14 is really an inspection of that indication. If there is such an indication (YES) then the indication is cleared and step 15 schedules a process RECEIVE FAX whose task it is to actually store the digitized form of the incoming fax into memory. Upon its conclusion RECEIVE FAX returns to step 12.

If step 14 determines that there is no new incoming activity that is a fax, then step 16 determines if a "complete" fax has been received. In the preferred embodiment this is accomplished by testing for an unlocked file in a directory (of disc files) used by the process RECEIVE FAX to store incoming fax messages. RECEIVE FAX does not create such an unlocked file until it has received some minimally useful quantity of information, say, an entire page of a fax message. (For our purposes here, such a partial fax message is "complete" even if parts of it are indeed missing. Just what to do about abnormal terminations resulting in missing parts is a matter of choice. By routing what did arrive, at least Charlie will know that somebody has tried to send him something, and further steps can now be taken by the parties.) If step 16 determines that such a "complete" fax has indeed been received step 17 schedules the process ROUTE FAX whose task it is to determine the E-mail address of the destination of the fax, and then mail the fax to that address. As the first step in that activity it locks the file so that it will cause no further instances of "YES" at step 16. Upon conclusion step 17 returns to step 12.

The remaining steps of the flow chart of Figure 2 deal with non-fax messages. Step 18, for instance, determines if there is incoming activity that is of interest, even though it may not be a fax message. If there is, step 19 schedules an appropriate process to handle it. We are speaking very generally here, as it is difficult to anticipate all the possibilities. However, a regular data communications transmission (e.g., over a Bell 102/103 compatible modem) is a good example. The appropriate process scheduled by step 19 returns to step 12 upon its conclusion.

If step 18 does not branch to step 19, then it branches to step 20 instead. Step 20 determines if there is outgoing activity that is to be initiated. If there is then step 21 schedules an appropriate process to accomplish that. That process returns to step 12 when it reaches its conclusion. Likewise, if step 20 determines that there is no outgoing activity, it also returns to step 12.

Refer now to Figure 3, which is a flow chart of the process IDENTIFY, scheduled in step 13 in Figure 2. After starting at location 22, the first step 23 is the determination of whether or not the activity detected in step 12 of Figure 2 (NEW INCOMING MODEM ACTIVITY) is the sending by the modem of the string "RING" (or some other equivalent activity). For the sake of explanation, assume that the modem is like one famous brand of modem currently on the market. That particular modem and its imitators send the string "RING" to its controller whenever the subscriber loop is experiencing a ringing current. The modem is pre-programmed to go off-hook after some predetermined number of rings, upon which the modem transmits another message to the controller (e.g., "CONNECT 2400" for a datacom connection, or perhaps "INCOMING FAX" for a fax).

If step 23 determines that the modem sent "RING", a count of the number of rings is maintained and step 24 used to determine if an abnormal number of rings is occurring. If there is, then something is amiss, and step 24 transitions to step 31, RESPOND APPROPRIATELY TO ILLEGAL CONDITION. But in a normal case of a pick-up on, say, the third ring, step 24 will be reached twice, after each of which it will transition back to step 23. Now the modem sends not "RING" but something else, having gone off-hook and

5 06-11-2004

Printed: 11-11-2004 EDOCP 02019563

established a connection on the start of the third ring. The "something else" causes step 23 to transition to step 25, where it is determined if the modem sent "INCOMING FAX" or not. If the determination is YES then step 26 assigns a value "FAX IN" to variable called STATUS that survives the death of the process. (This particular mechanism comes from the UNIX brand operating system; the actual code for the preferred embodiment was written in C and runs under the UNIX brand operating system. Other inter-process communication mechanisms could be used.) The value of STATUS is passed back as a parameter to the process that scheduled IDENTIFY in the first place. Step 27 is a normal exit back to the calling program.

If step 25 determines that the modem did not send "INCOMING FAX" step 28 determines if it was some other legitimate response, such as "CONNECT 1200" or "CONNECT 2400". If the determination is NO, then something is amiss, and the transition is to step 31 for an appropriate response to an illegal condition, followed by an exit back to the calling program. Otherwise, step 28 transitions to step 29, which schedules the appropriate (and legitimate) process. Following that there is a normal exit at step 30 back to the calling program.

Figure 4 is a simplified flow chart of the process RECEIVE FAX scheduled by step 15 in the flow chart of Figure 2. At the level of explanation that suits our present purpose the process RECEIVE FAX may be thought of mainly as a single step 34 following a start location 33. That step 34 is to receive the fax and store it in an unlocked file in a directory (of disc files) whose contents are incoming fax messages. Following this the process terminates itself at exit step 35. Naturally, the program corresponding to step 34 is susceptible of subdivision into several steps, some of which are themselves divisible. But on the whole, the step 34 is conventional and well understood in the art, and we needn't discuss it further.

Turn now to Figures 5A-B, which are a flow chart of the process ROUTE FAX that is step 17 in Figure 2. (A program listing for ROUTE FAX appears in Appendix II.) The first step 37 in the process ROUTE FAX is to lock the file in the subdirectory containing the received image. This action keeps the file from being processed more than once by step 17 in the flow chart of Figure 2. In the next step 38 a page pointer is set to point to the first page of the document and a header pointer is set to point to the first header. The headers are bar coded information expected to appear on the cover sheet in accordance with the invention. (Setting a pointer to the first header does not locate that header, but instead indexes where to store the header information once the first header is located and processed.) The next step 39 is to set a page orientation flag to a value meaning "unknown."

Steps 40-48 constitute a loop for detecting bar coded headers and converting the bar codes to ASCII, both done in preparation for later attaching the ASCII header information to the body of the message. To this end, step 40 fetches the next (or first) raster line (scan line) of the first page of the fax. No header can precede this location, although it is possible that one could occur this early in the document. Step 41 partially decodes the line fetched in step 41. What goes on here is this. The CCITT transmission is a string of sequences of ones and zeros representing symbols in a compression scheme. The partial decoding trades the sequences representing symbols in the compression scheme for a simple run length encoding of black and white pixels, beginning with white. Step 42 is the application of a procedure named BAR CODE, one of whose jobs it is to determine if the fetched raster line is a line of bar code. The procedure BAR CODE is described in connection with Figures 6A-B. Step 43 tests the outcome of the procedure BAR CODE; in the simplest case the fetched raster line was not bar code, whereupon step 44 determines if there are yet more lines for this page of the document. If there are the loop continues from step 40. Otherwise, there can be no further headers, and the loop of steps 40-48 is complete, leading to the execution of step 49 (shown on Figure 5B and discussed a little further on below).

To return to step 43, if the fetched raster line was indeed bar code, steps 45-48 cooperate to store its content (the legend) as the current header. To this end, step 45 inquires if the line or segment of bar code in hand is a duplicate of the last line or segment of bar code from steps 42 and 43. (Upon initial entry the answer will be NO, since a previous legend has not yet been stored.) Unless the answer is NO, it not desirable to further process the bar code, as it is probable that the raster line in hand is simply a subsequent scan line of a multi-scan-line image for a bar code already processed (with an earlier fetched scan line). If this is the case then this scan line is to be ignored. Assuming the answer is NO, then step 46 determines if the line in hand is an initial or first segment of a legend. (That is, it is a logical line of only one physical line or is the first physical line of a logical line having several physical lines. This is the same as having a LINE OF cipher that is any of the "one of" characters described in connection with Figure 7.) If the answer is YES then step 46.1 clears the legend buffer, which is the temporary location for storing and assembling the legend into a complete header. Otherwise, step 47 determines if the physical line of bar code in hand is a valid continuation of the previous one. That is, if the previous line was number one of three, is this one number two of three, and if the previous one was number four of six, is this one number five of six, etc. If it is a valid continuation, then the loop continues with step 46.2, which appends the

6 06-11-2004

Printed: 11-11-2004 EDOCP 02019563

decoded ASCII string of the line or segment in hand to the legend buffer. Otherwise, the line in hand is of no interest, and is ignored by continuing with step 44.

From step 46.2 the loop continues with step 48, which determines if the line or segment in hand is the last segment (is the last physical line in the logical line). That is, is the LINE OF cipher of the form "one of one", "two of two", "three of three", etc. If it is, then step 48.1 stores the legend buffer into the header location currently pointed at, and step 48.2 increments the pointer to point at the next header. Following that, the loop continues with step 44. Otherwise, steps 48.1 and 48.2 are skipped, and the loop continues with step 44 directly.

After the entire first page has been scanned and all headers detected and stored, step 49 determines if the first header is right side up or upside down. If the first header is upside down then it seems probable that the entire document is also upside down (although it is possible that this unfortunate condition applies only to the first page), and step 50 turns the entire document over (i.e., right side up). If the first header is already right side up step 49 skips step 50 and continues instead with step 51.

The ability to determine if the first (or for that matter, any) header is right side up or upside down arises from the asymmetrical nature of a certain "START/STOP" cipher occurring at distal ends of a physical line of bar code and used to enclose bar coded information. The idea is that if the document is right side up (and by implication the first page with its first header, too) then a left-to-right scan of the first header will be successful in detecting the enclosing "START/STOP" ciphers. If the header (an by implication, the rest of the document) is upside down, then a left-to-right scan will be unsuccessful, but a right-to-left scan will succeed.

Step 51 packs the fax message into ASCII characters as a format for transmission over the network 8. That is, at this point the fax consists of a file of sequences of ones and zeros representing a compressed (in the sense of data compression) image, of the same type to which the partial decoding of step 41 pertained. These sequences are not as conveniently transmitted over the network 8 as are ASCII characters having a regular predefined length. So, the bits of the sequences are simply taken in six bit chunks and added to an offset, so that they may be treated as though they were printable ASCII characters. Any framing control bits, start and stop bits, parity bits, etc., that are then added for transmission on the network 8 are then later removed by the software at the destination, as is the offset used to position the six bit pattern into the range of printable characters, and the transported groups of six bits are reassembled into their original sequences. Software at the destination can then proceed to convert the received fax into a displayable or printable image.

Following step 51, step 52 attaches the ASCII characters representing the actual content of the headers to a preamble segment in the E-mail package containing the fax message. This is one of a plurality of uses that are made of the ASCII representation of the content of the headers, as now explained. The actual bar coded header is itself a visual image of some bar code, and is transmitted as any other graphical component of the fax message (i.e., as described in the preceding paragraph). What step 52 does is to decode the bar code of each header and supplement the graphical description of the image of the header with actual ASCII codes representing the content of the header. These ASCII strings are placed into predefined places for such information located in a preamble portion of the E-mail format. (This is similar to pads of memo paper having already printed thereon the headings "TO:", "FROM:" and "SUBJECT:".) This makes the header content readable even though the fax itself has perhaps not yet even been rendered as a graphic image by the fax viewer. That is, a person looking into the beginning of a file containing the stuff the fax viewer is going to use can see the ASCII content of the headers. (See the TO: line on page 1 of Appendix V.) Also, step 54 (described in the next paragraph) needs the decoded content (meaning) of the "TO:" header for routing purposes. And, upon reflection, it will be appreciated that even when the fax is finally rendered by the fax viewer into an image, or a series of images, the headers would then appear only as bar code, just as they appeared on the originally transmitted document. Who likes to read bar code, anyway? Fortunately, there is no need to learn how to read the bar code, as the sender can use a program for header page creation (MAKE_HEADER, described later) that not only produces the desired bar code image but also creates adjoining that bar code the graphic image of the corresponding ASCII characters. Thus a header page can have not only bar code readable by the fax server but also corresponding alphabetic characters that are readable by a human being.

Next, step 53 determines if there is a TO: header. If there is its ASCII content is used in step 54 to create the ASCII representation of the E-mail destination address that the bar code of the TO: header represents, as shown in Appendix V. If there is not step 55 sets the E-mail address to some default value, say, general delivery. Either way, the next step 56 mails the fax. Following that the process ROUTE FAX terminates with the exit at step 57.

The astute reader will note a similarity between steps 52 and 54/55. It may appear that step 54 is

7 06-11-2004

Printed: 11-11-2004 EDOCP 02019563

redundant, having been accomplished in step 52. Whether or not this is so may depend upon the inner workings of the E-mail system. In the system we used, step 54/55 was necessitated by the requirement by the E-mail system that it be expressly passed a destination parameter at the time it is called, regardless of what good information may be in the header portion of the file or package being mailed.

Refer now to Appendix IV and Appendix V. Appendix IV show the binary format of a file containing the image of the faxed document. This file format permits arbitrary binary data, and as such may not be easily transmitted over some networks since many of the patterns that would occur would not correspond to benign ASCII characters. Note also that this format does not include any header information that pertains to the origin and destination of the fax itself. The fax itself is in there, but the characters that would result from decoding the raster lines and assembling them are not apparent.

Appendix V illustrates a sample message that can be transmitted over the network, and that once was in the form of Appendix IV. Two things have happened between Appendix IV and Appendix V. First, the arbitrary binary format of Appendix IV is brought under control by application of the UNIX utility uuencode. This renders the arbitrary binary file as a sequence of printable ASCII characters. Thus, Appendix IV maps into that portion of Appendix V following "begin 444 fax". Second, above that point in the format of Appendix V is the E-mail header stuff. This is the place where the TO: line in the faxed document, decoded by instances of steps 42 and 43 in Figure 5A and stored by steps 52 and 54 in Figure 5B, gets placed into the message routed by the E-mail system. (Look under the X-FAX-FROM line, which now stands for "FROM:", since the E-mail system insists that its FROM be for the entity that gave it the message to mail, which in this case is the fax server (Fax Daemon).) To drop the other shoe, once the mailed fax gets to the fax viewer (whose source code is shown in Appendix VI) the UNIX utility uudecode is used on the fax image portion to recover the binary data representing the image.

Refer now to Figures 6A-B, which are a flow chart for the procedure BAR CODE appearing as step 42 in Figure 5A. (Appendix III contains a program listing of the procedure BAR CODE.) Recall that one of the purposes of the procedure BAR CODE is to determine if a scan line contains bar code or not.

Following a start represented by location 58, the first step 59 in the procedure BAR CODE is to determine if there are at least enough transitions (white-black or black-white pixel pairs) in the line for it to represent bar code in the first place. If the answer is "NO" then the procedure has found the desired result (the scan line is not bar code) and continues with step 70, which sets a return value to "NOT BAR CODE". This is followed at step 71 by a RETURN to the calling process. The return value mentioned above is a parameter passing communication device found in the C programming language. It is used here as a way to inform a calling routine about the now concluded goings-on in a procedure that was called.

If step 59 determines that there are indeed a sufficient number of transitions in the line, then it is possible that the line contains bar code, although at this point it is by no means a certainty that it does. After the "YES" transition from step 59 the next step 60 clears the reverse flag. This flag indicates which direction of scanning to use. For our purposes here we may say that if the reverse flag is cleared, then forward scanning of from left to right is in use. If the reverse flag is set then reverse scanning of from right to left is in use. So, step 60 initially sets forward scanning.

The next step 61 determines the value of the page orientation flag. It will be recalled that this flag was initially set to a value of "UNKNOWN" by step 39 in the flow chart of the process ROUTE FAX (Fig. 5A). Thus, upon initially reaching step 61 of Fig. 6A the answer to the question "IS THE PAGE ORIENTATION FLAG = UPSIDE DOWN?" will be "NO" and the procedure BAR CODE will continue with step 64.

Step 64 is a fast search in the selected direction for a START cipher. A START cipher is a START/STOP cipher that occurs at the beginning of a line, and a STOP cipher is a START/STOP cipher that occurs at the end of a line. Since the same physical cipher is used for both START and STOP; which one is which is then distinguished by its location relative to the interior portion of the line. Also, separate ciphers could be used for START and STOP, although this is not necessary. Step 65 determines if a START cipher was found. If the answer is "YES" then step 66 performs a similar fast search in the selected direction for a STOP cipher. Next, step 67 determines if the STOP cipher was indeed found. If the answer to either of steps 65 or 67 is "NO" then the procedure BAR CODE continues with step 68, which determines if the page orientation flag is set to a value of "RIGHT SIDE UP". That is, if a previous application of BAR CODE has decided that the page is indeed right side up, then the current scan line must not contain bar code. Accordingly, upon a "YES" answer at step 68 the procedure continues with step 70 (already explained) followed by the RETURN at step 71.

If the page actually is upside down, however, then searching for the ciphers from left to right would fail to find them, and the answer at step 68 would be "NO" (the page orientation at this point could be either upside down or unknown). Of course, the ciphers might simply not be there to be found, either. But that can't be known until after searching for them from right to left. Under these conditions the next step 69 is to

8 06-11-2004

Printed: 11-11-2004 EDOCP 02019563

ask if the reverse flag is set. If it is, then we have already checked the other direction of scanning, and can now say that the scan line does not contain bar code. Therefore, a "YES" answer at step 69 also produces a transition to steps 70 and 71. However, a "NO" answer at step 69 means that there is still a chance that the document is upside down. Under these conditions the procedure continues with step 62, followed by step 63. These steps reverse the scanning direction and set the reverse flag. Following this the search for START and STOP ciphers is repeated. Unless both are found then we again find ourselves at step 68. The answer will be "NO" (if it were "YES" it would also have been "YES" on the first pass, and there would not have been a second pass....) and the procedure resumes with a second pass through step 69: "IS THE REVERSE FLAG SET?" After the second pass it will be, and the attempt to find bar code in the scan line has failed. That leads to steps 70 and 71, which indicate the failure to find bar code and terminate the procedure.

We can now consider the case when the answer to step 67 is "YES". This means that both a START and STOP cipher were found for some direction of scanning; the reverse flag indicates which. The procedure then continues with steps 72 and 73. Step 72 computes the number N of ciphers between START and STOP, and step 73 sets a pointer to the first cipher after START.

Steps 74-78 are a loop that trades ciphers for ASCII characters. To this end step 74 determines from the bar code transition data a code representing the associated cipher. Step 75 uses that code to index into a table of ASCII values for the ciphers. Step 76 gets the ASCII code or an error code and stores it into the next position in a work string. Step 77 determines if there are more ciphers to convert, and if there are ("NO") then step 78 points to the next cipher and closes the loop by continuing at step 74.

If all the ciphers have been converted the answer at step 77 will be "YES", and the procedure continues with step 79: "IS CHECKSUM OK?" If the answer is "NO", then something has gone amiss and the situation is treated as though the scan line is not bar code. This is accomplished by continuing with steps 80 and 81, which correspond to steps 70 and 71.

However, if the checksum is OK, then good bar code is in hand and the procedure continues with steps 82-85. Step 82 sets the value of the page orientation flag according to the value of the reverse flag. The idea here is that the process described above aligns the reverse flag with whichever assumption about page orientation first produces successful bar code. Step 82 assigns a value to the page orientation flag that corresponds to the first successful value of the reverse flag. There is an implicit latch that allows the page orientation flag to be determined only once, so that subsequent assignments of value to the page orientation flag at step 82 are always of the same value as the first assignment.

Consider a subsequent application of BAR CODE for a header after the first one. Step 60 will clear the reverse flag, but now the page orientation flag will have a value of RIGHT SIDE UP or UPSIDE DOWN; UNKNOWN will not be a choice. If the document is right side up then steps 62 and 63 will be skipped, leaving forward scanning in effect. Presumably, steps 64-67 will be successful, avoiding any attempt to change the value of the reverse flag, so that at step 82 no change occurs to the page orientation flag, since it already has the value RIGHT SIDE UP.

If the document is upside down at the start of the subsequent use of BAR CODE, the answer at step 61 will be "YES", which allows steps 62 and 63 to reverse the scanning direction and set the reverse flag. Presumably, the searches and checks of steps 64-67 will now be successful, leaving step 82 to again assign a value of UPSIDE DOWN to the page orientation flag.

If steps 64-67 are not successful, then the answer at step 68 will be "YES" if the page orientation flag has already been set to RIGHT SIDE UP, transitioning to the unsuccessful exit of steps 70 and 71. The logic here is this: The page orientation flag said at step 61 that the document is right side up, so it was scanned from left to right. But the scanning failed. So, if the document is supposed to be right side up at step 68, then the line must not be bar code, and the update operation for the page orientation flag is never even reached. The remaining case is where step 61 says the document is already upside down, so steps 62-63 reverse the scanning direction and set the reverse flag. But now steps 64-67 fail. The answer at step 68 will be "NO", but since the reverse flag was set in step 63, step 69 will transition to the failure exit of steps 70-71. That averts any attempt to assign another value to the page orientation flag at step 82. Here the logic is: The page orientation flag said at step 61 that the document was upside down, so step 63 set the reverse flag. Thus, if the cipher-finding operation fails (steps 64-67) and the reverse flag is already set (step 69), it must not be bar code.

To conclude the procedure BAR CODE, step 83 saves the type (TO:, FROM:, etc.) and ASCII content (the work string) of the header. Then step 84 sets the return value to "BAR CODE", and step 85 does the RETURN that terminates the procedure.

To this point we have been examining what could be termed an overview of the operation of the fax server. We turn now to a discussion of certain issues pertaining more directly to the bar code preferred for

9 06-11-2004

Printed: 11-11-2004 EDOCP 02019563

use with the fax server. These include the layout of the header page, a description of the preferred bar code itself, and a means for generating images of bar code.

Refer now to Figure 7, which is an illustration of a line of bar code 86 adapted for use on a header page intended to cooperate with the fax server 7 of Figure 1. The particular header line shown is for a TO: header; it could as easily have been for any of the other types of headers. The particular line 86 shown is composed of fifteen bar code ciphers and represents the information shown in the corresponding ASCII header line 87.

The line of bar code 86 begins with a START/STOP cipher 88 and ends with a START/STOP cipher 93. Every physical line of bar code begins and ends with such START/STOP ciphers, regardless of whether or not it is part of a logical line composed of several physical lines. The second cipher in the line 86 (or in any physical line) is a TYPE cipher 89. This cipher indicates what kind of logical line (i.e., what its function is, TO:, FROM:, etc.) this physical line belongs to. The various TYPES may include:

TO:	FROM:	SUBJECT:
XXX:	YYY:	ZZZ:

The X's, Y's and Z's in the line above represent the possibility of defining other types to serve particular functions. For example, a TYPE cipher might be construed to mean "first quarter '90 bingo card", which would implicitly indicate a particularly formatted request for information.

In addition to all the types mentioned to this point is a type we may call "NCT" (No Coded Type). When a header has type NCT it means that the header does indeed have a type, but rather than being indicated directly by the cipher the type is spelled out by the ASCII symbols in the content portion of the header. To extend the bingo card example above, one might find the NCT cipher followed by "IQ90BINGO".

Although our preferred method of identifying header type is by the use of a TYPE cipher as explained above (either an expressly predefined type or the NCT type followed by a definition in the content portion), it is entirely conceivable that the notion of a type indicating cipher could be dispensed with in favor of simply including the keywords TO:, FROM:, etc. in the content portion. The use of the START/STOP ciphers and the LINE OF mechanism (described below) would still work as before. Now the problem becomes one of parsing the content portion to determine what use to make of it.

The third cipher in any physical line is a LINE OF cipher 90. This cipher indicates how many physical lines comprise the logical line, and which one of that number this particular physical line represents. In the case where there is but one physical line in the logical line the LINE OF cipher 90 will be a bar code pattern whose meaning is "one of one". If a logical line has two physical lines then one of those two will have a LINE OF cipher whose meaning is "one of two" and the other one will have a meaning of "two of two". The other meanings are "one of three", "two of three", ... "four of four", etc. Since there are 150 different values for a cipher in the preferred bar code, the preceding list goes as high as sixteen of sixteen (not enough ciphers are available to complete the "of seventeen" series).

An extension mechanism is provided to allow the number of physical lines in a logical line of bar code to be greater than sixteen. This works as follows. One of the ciphers falling into what would otherwise be part of the "of seventeen" series is selected to indicate that the next two ciphers M and N indicate "M of N", bringing the number of possible physical lines up to a full one hundred and fifty.

Following the LINE OF cipher 90 occur zero or more ciphers 91 representing the "actual content" of the line. In the example shown in the figure this is the (fictitious) string "jsm@hpfcrp". After the content string 91 occurs a CHECKSUM cipher 92, which is in turn followed by the STOP cipher 93.

The line of bar code 86 and the line of ASCII characters 87 in Figure 7 were produced by a program called MAKE_HEADER. The source code for the program appears in APPENDIX I. MAKE_HEADER accepts as input an indication of what type of line to produce, along with any content string. The output is produced upon a fax machine or upon a suitable graphics output device, say a display monitor or a high resolution printer. The height of the resulting line of bar code is selectable, from a minimum of one pixel to as high as the output medium can produce. What height buys is immunity to skew, and since most fax machines have pretty good paper transport mechanisms, a minimum height of one half of an inch is plenty, even for lines of maximum width. It was found that most machines would jam the paper before continuing to feed a sheet with sufficient skew to disturb the reading of full width lines of bar code only one quarter of an inch high.

The widths of the wide and narrow lines and of the wide and narrow spaces between the lines are not fixed at any particular size, although there are preferred ratios amongst them. Within the limits of

10 06-11-2004

Printed: 11-11-2004 EDOCP 02019563

practicality, the MAKE_HEADER program can make the ciphers wider or narrower, as desired. In cooperation with this, the bar code reading capability in the fax server does not expect set sizes of bar code ciphers, understanding instead what the preferred ratios are, and accepting (within practical limits) arbitrarily scaled ciphers. Accordingly, the length of a line of bar code is not rigidly cast in advance, either, except that a maximum physical length can be specified to respect output device limitations. Thus, a given logical line might fit on a single physical line if the cipher width is set at or below a certain size, but will automatically be broken and reformatted into two (or more) physical lines either as the number of characters in the content string increases or as the selected width of the ciphers is increased.

Figures 8A-D illustrate what has been discussed above. These figures represent four different instances of the same header page. (Other useful stuff that might be found on an actual header page, such as company logos, reply phone numbers, page count, time and date have been omitted for clarity. Those things would not affect the operation of the invention, since they are not bar code, and they can appear in any convenient locations.) Each of these four figures is logically equivalent to the other three, although they are each different in various physical respects. In particular, note that in Figure 8A all three of the TO:, SUBJECT: and FROM lines fit on respective single physical lines. In the following figures various ones (or all) of the logical lines are broken into instances of multiple physical lines. A careful inspection of those instances will reveal the existence of START and STOP ciphers for each physical line, as well as the different TYPE and LINE OF ciphers.

Refer now to Figure 9, which is an illustration of the bar code preferred for use with the invention. The figure shows one hundred and fifty ciphers, of which a portion 94 of one hundred and twenty eight represent either the ASCII character set or the ordinal numbers associated with the "line of" mechanism for multi-line bar codes. The remaining portion 95 of twenty-two ciphers are available for control codes, as well as for use as ordinals in the multi-line mechanism. At the left-hand side of the portion 64 is a column of hexadecimal addresses. Each address indicates a starting address for the left-hand edge of the associated row in the portion 94. The individual ciphers in the portion 94 have addresses that increase from left to right along rows, and increase from top to bottom by rows, as indicated by the arrows 98. The hexadecimal addresses of the ciphers identify the corresponding ASCII character according to the arrangement depicted in Table I.

Finally, note ciphers 96 and 97; these are the START/STOP cipher. Cipher 96 is the START/STOP cipher as scanned in the forward direction, and cipher 97 is the same cipher when scanned in the reverse direction.

TABLE I

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0a	nl	0b	vt	0c	np	0d	cr	0e	so	0f	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	can	19	em	1a	sub	1b	esc	1c	fs	1d	gs	1e	rs	1f	us
20	sp	21	!	22	"	23	#	24	$	25	%	26	&	27	'
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c	|	7d	}	7e	~	7f	del

One conventional bar code in wide use today is the "code-39" bar code. The name arises from the use of three wide elements (two black and one white) out of a total of nine. The problem with code-39 is that it has only ninety-some ciphers, while at least one hundred and twenty-eight are need for a full implementation of ASCII, with preferably a few more ciphers available for use as control characters. The bar code depicted in Figure 9 may be termed "code-411", as it uses eleven total elements, of which four are wide (two white and two black) and seven are thin (three white and four black). Each cipher starts and ends with

06-11-2004

Printed: 11-11-2004 EDOCP 02019563

black, and elements alternate between black and white. Ciphers may be separated by an arbitrary amount of white.

Table II lists, according to an arbitrarily selected algorithm for ordering the ciphers, the binary coded hexadecimal representation of the code-411 bar code (the "Ox" is the convention in the C programming language for denoting hexadecimal integers, and is not itself part of the bit pattern). In the hex representation a one denotes a wide element (whether white or black) and a zero denote a thin element. Replace each of the three hex digits with its binary equivalent, and take the right-most eleven bits of the twelve bits. The most-significant bit of the resulting hex code corresponds to the left-most element of a cipher that is right side up. It will also be necessary to recall that each cipher begins and ends with black elements, and that black and white elements occur in strict alternation.

TABLE II

```
0x780, 0x720, 0x708, 0x702, 0x5a0, 0x588, 0x582, 0x528, 0x522, 0x50a
0x6c0, 0x660, 0x648, 0x642, 0x4e0, 0x4c8, 0x4c2, 0x468, 0x462, 0x44a
0x690, 0x630, 0x618, 0x612, 0x4b0, 0x498, 0x492, 0x438, 0x432, 0x41a
0x684, 0x624, 0x60c, 0x606, 0x4a4, 0x48c, 0x486, 0x42c, 0x426, 0x40e
0x681, 0x621, 0x609, 0x603, 0x489, 0x483, 0x423, 0x40b, 0x3c0, 0x360
0x348, 0x342, 0x1e0, 0x1c8, 0x1c2, 0x168, 0x162, 0x14a, 0x390, 0x330
0x318, 0x312, 0x1b0, 0x198, 0x192, 0x138, 0x132, 0x11a, 0x384, 0x324
0x30c, 0x306, 0x1a4, 0x18c, 0x186, 0x12c, 0x126, 0x10e, 0x381, 0x321
0x309, 0x303, 0x1a1, 0x189, 0x183, 0x129, 0x123, 0x10b, 0x2d0, 0x270
0x258, 0x252, 0x0f0, 0x0d8, 0x0d2, 0x078, 0x072, 0x05a, 0x2c4, 0x264
0x24c, 0x246, 0x0e4, 0x0cc, 0x0c6, 0x06c, 0x066, 0x04e, 0x2c1, 0x261
0x249, 0x243, 0x0e1, 0x0c9, 0x0c3, 0x069, 0x063, 0x04b, 0x294, 0x234
0x21c, 0x216, 0x0b4, 0x09c, 0x096, 0x03c, 0x036, 0x01e, 0x291, 0x231
0x219, 0x213, 0x0b1, 0x099, 0x093, 0x039, 0x033, 0x01b, 0x285, 0x225
0x20d, 0x207, 0x0a5, 0x08d, 0x087, 0x02d, 0x027, 0x00f, 0x429, 0x4a1
```

When produced by the program MAKE_HEADER (APPENDIX I) certain default parameters are used. These include:

Thickness of wide black = .0100"
Thickness of wide white = .0133"
Thickness of thin black = .0033"
Thickness of thin white = .0066"
Intercipher spacing = one thin white
Height of bar code elements = .50"

In addition, the START/STOP cipher (as produced by MAKE_HEADER) is fifty percent larger in all dimensions except height. This is not an absolute requirement; it is done simply as a safety measure to ensure added reliability.

The above default values were chosen to facilitate reliable readability after fax - photocopy - fax cycles. Both faxing and photocopying tend to cause changes in black widths, even if there is no enlargement or reduction in the overall size of the document as a whole. The default height was selected to allow as much as three and one half degrees of skew in a received document of eight and one half inches in width. Our experiments suggest that the nature of the paper transport mechanism in most fax machines limits even deliberately induced skew to approximately three degrees.

Refer now to Figure 10, wherein is depicted in schematic form one way that a "bingo card" 99 could be arranged for use with the invention. As indicated in the figure, one or more lines of bar code 100 identify the bingo card to the fax server. An array of check boxes 101 is located on the card. These may be in a known spatial relationship to suitable targets 105 and 106, or perhaps simply in fixed relation to the bar code 100. It is also possible that the edges of the check boxes in the array are independently recognizable, and that the check boxes can therefore be arbitrarily located. Associated with each check box is a field of text 102. Each field of text contains a human readable explanation of what is to be understood by placing a mark inside the associated check box. For example, if the bingo card 99 were a request for literature, each field of text 102 would contain the title or description of a piece of literature that could be requested. An optional encoding matrix of check boxes may be included to allow the sender of the card to convey other information, such as a reply phone number. In that particular example, a ten by ten array of check boxes could serve to encode an area code followed by a seven digit phone number. In many instances this would

12 06-11-2004

Printed 11-11-2004 EDOCP 02019563

be unnecessary, as the requested information could simply be faxed back as part of the same phone call, relieving the fax server from even needing to know the phone number of the caller. An optional arrow 104 indicates a direction to feed the image of the bingo card 99 through the fax machine, in the event there is some advantage to this. What was said earlier about document erection still stands, but absent some instruction like arrow 104 the fax server may be forced to store and manipulate an entire graphics image of the bingo card if sideways or angled cards are faxed. The information is still all there, but now the scan lines of the fax machine are not essentially parallel to the "axis of information progression" in the document, and more complicated algorithms would be required.

The image of the bingo card 99 is distributed in any convenient fashion; e.g., in a magazine. To use the card the respondent fills it out and either places it in a plastic sleeve or photo copies it onto full size paper prior to faxing. Then he faxes it to the number indicated.

The faxable bingo card is not limited in its application to situations involving a static or predetermined response, such as replying with a prewritten document. It may also be used in conjunction with replies that are custom generated by the fax server (or other application at the destination) at the time the request or other transaction is undertaken. Say, for example, an account holder wishes a printed statement indicating the status of his account. Or consider, as a second example, a case where the bingo card is filled out and faxed in order to place an order for merchandise. Upon receipt of the order the fax server immediately prepares and sends an order acknowledgment, confirming what ordered and indicating the billing arrangements. While remote ordering systems can be based on tones transmitted from a telephone keypad, they are awkward with regard to verification and error correction. The bingo card has the advantage that it can be filled out, studied for correctness, and altered if need be. What is more, the faxing of the bingo card is, as far as the user is concerned, a unitary operation once the proper phone number is dialed. There is no worry about accidentally pressing the wrong key during the order entry process, or having a key produce either a missing or double burst of tone as sometimes happens when the finger is not placed squarely in the center of the key, resulting in a binding action within the key.

Claims

1. A method of addressing a document to be transmitted to an addressable destination on a network, the method comprising the steps of:

   expressing the address of a destination on a network as a visual bar coded image upon a page of a document to be transmitted by facsimile;

   encoding the document according to a format for the digital transmission by facsimile; and

   sending the encoded document to a fax server connected to a network upon which the destination address is located.

2. A method of routing the digitally encoded facsimile representation of a document to a destination on a network, the method comprising the steps of:

   scanning a selected page of the document for bar code;

   detecting the presence of a selected bar code cipher in a selected location within the scanned bar code;

   converting a remaining portion of the scanned bar code into an address suitable for use in addressing a destination location on the network; and

   sending the document to the destination location on the network.

3. A method as in claim 2 wherein the selected cipher is asymmetrical when scanned from opposing directions, wherein the scanning step is performed in opposing directions, and further comprising the step of erecting the pages of a document that is upside down in response to the detecting step being successful upon a bar code scanned in a selected direction.

4. A method of addressing a document to be transmitted to an addressable destination on a network, the method comprising the steps of:

   expressing the address of a destination on a network as a visual bar coded image upon a page of a document to be transmitted by facsimile;

   encoding the document according to a format for the digital transmission by facsimile;

   sending the encoded document to a fax server connected to a network upon which the destination address is located;

   scanning a selected page of the document for bar code;

13 06-11-2004

Printed: 11-11-2004 EPODOC 02019563

detecting the presence of a selected bar code cipher in a selected location within the scanned bar code;

converting a remaining portion of the scanned bar code into an address suitable for use in addressing a destination location on the network; and

sending the document to the destination location on the network.

5. A method as in claim 4 wherein the selected cipher is asymmetrical when scanned from opposing directions, wherein the scanning step is performed in opposing directions, and further comprising the step of erecting the pages of a document that is upside down in response to the detecting step being successful upon a bar code scanned in a selected direction.

6. A method comprising the steps of:

storing a digital representation of a document within a computer system at a destination ;

transmitting from a sending entity a bar coded request by facsimile to the computer system at the destination;

reading the bar code to determine an action to perform in response to the transmitting of the bar coded request; and

sending by facsimile the digital representation of the document to the sending entity.

7. A method as in claim 6 wherein the transmitting step includes a bar coded return address, and the sending step uses the bar coded return address in sending the digital representation of the document.

8. A method comprising the steps of:

storing digital representations of a plurality of documents within a computer system at a destination;

transmitting by facsimile from a sending entity a form including a bar coded request for the return transmission of information, indicia corresponding to at least one of the documents in the plurality thereof, and a bar coded return address accessible by facsimile;

reading the bar coded request for the return transmission of information;

determining from the indicia at least one digital representation to be sent to the return address; and

sending by facsimile to the return address each digital representation determined in the determining step.

9. A method as in claim 8 wherein the indicia are formed by the application, prior to the transmitting step, of marks within predefined fields indicated upon the form.

10. A method as in claim 9 wherein the predefined fields indicated upon the form are associated with human readable labels each specifying a document in the plurality thereof.

11. A method comprising the steps of:

transmitting from a sending entity a bar coded request by facsimile to a computer system at a destination;

reading the bar code to determine an action to perform in response to the transmitting of the bar coded request;

subsequent to the reading step, generating in the computer at the destination a digital representation of a document relating to the action to be performed, and

sending by facsimile the digital representation of the document to the sending entity.

14 06-11-2004

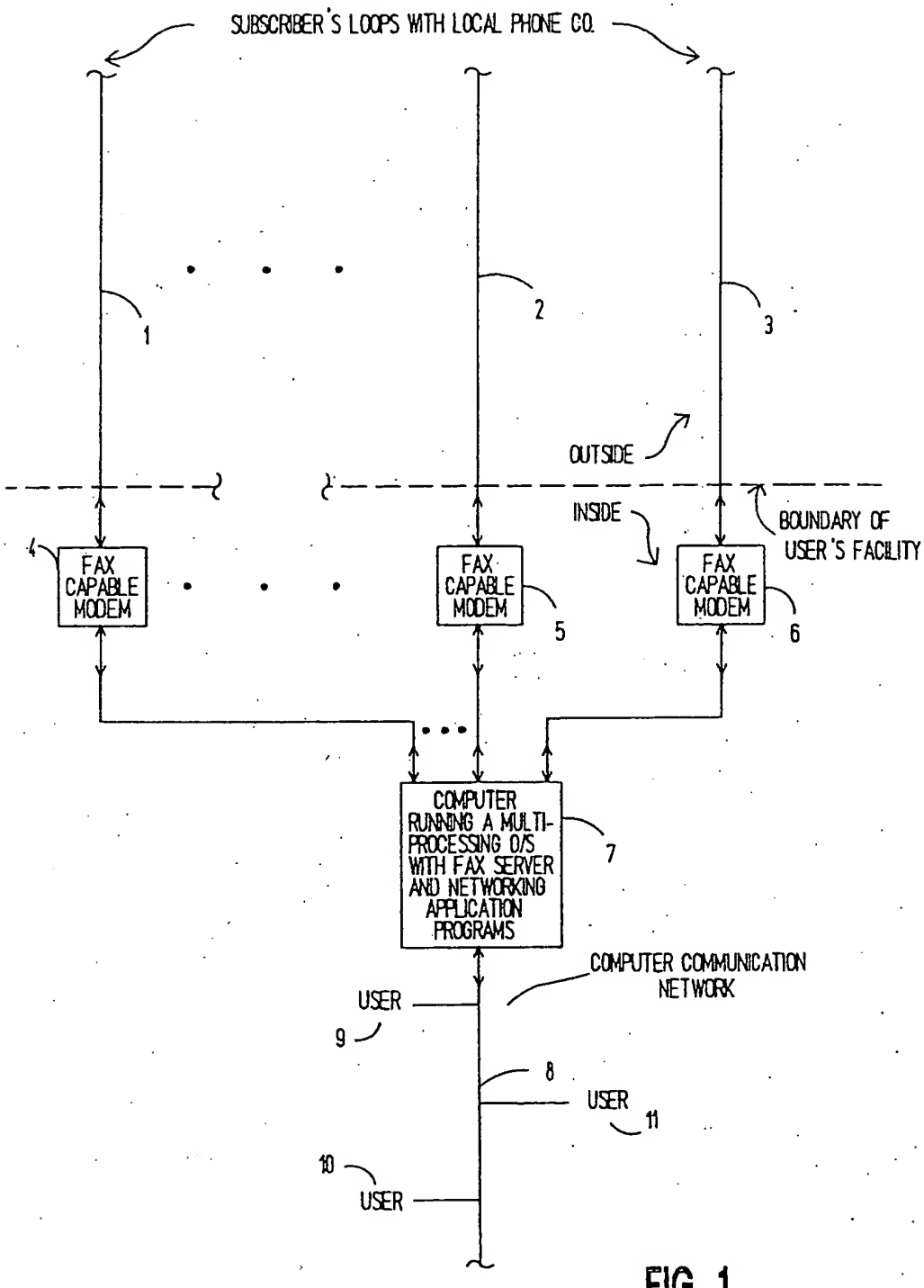
SUBSCRIBER'S LOOPS WITH LOCAL PHONE CO.
1
2
3
OUTSIDE
INSIDE
BOUNDARY OF USER'S FACILITY
4
FAX CAPABLE MODEM
FAX CAPABLE MODEM
5
FAX CAPABLE MODEM
6
COMPUTER RUNNING A MULTI-PROCESSING O/S WITH FAX SERVER AND NETWORKING APPLICATION PROGRAMS
7
COMPUTER COMMUNICATION NETWORK
USER
9
8
USER
11
10
USER

FIG 1

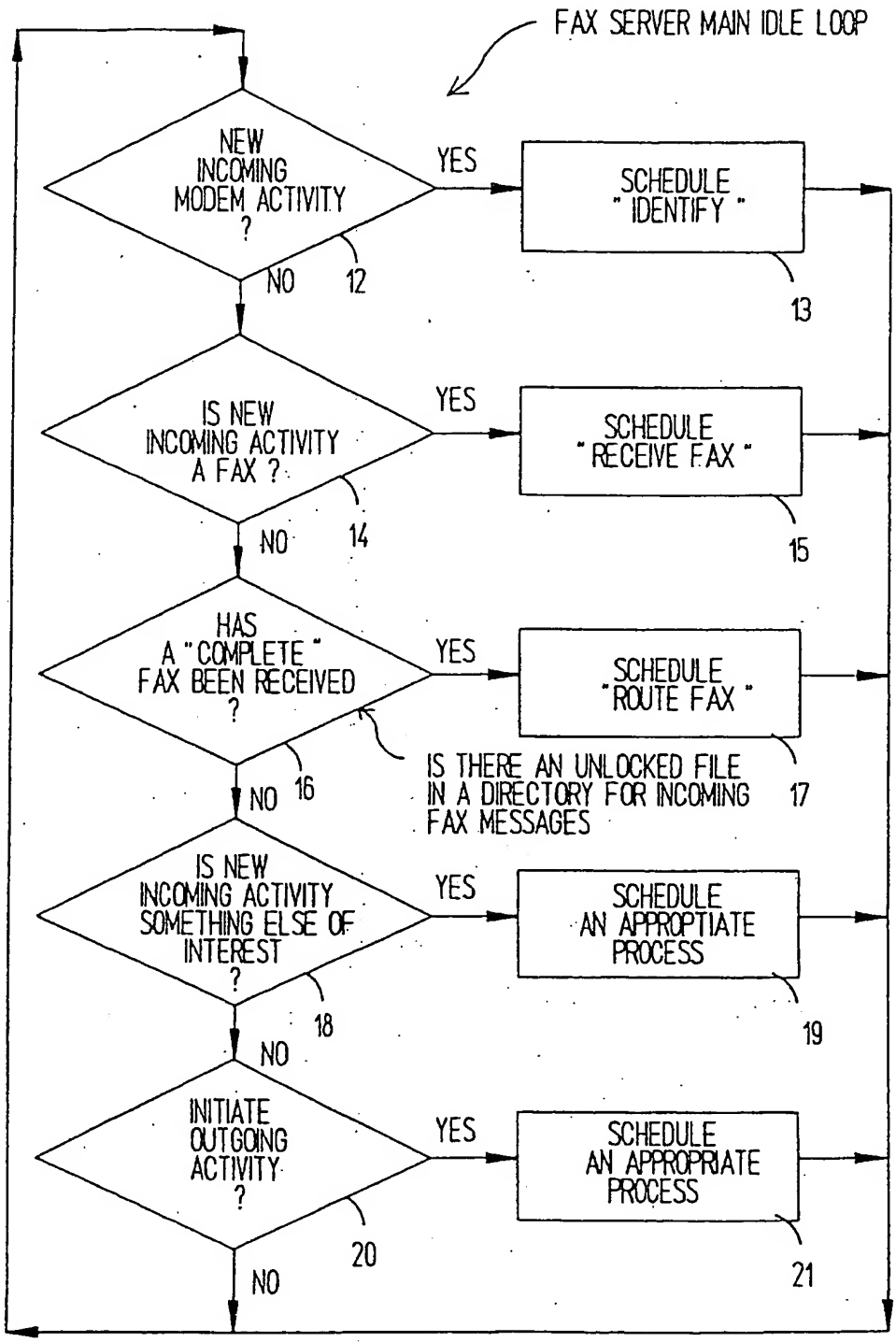
FAX SERVER MAIN IDLE LOOP
NEW INCOMING MODEM ACTIVITY ?
YES
SCHEDULE "IDENTIFY"
13
NO
12
IS NEW INCOMING ACTIVITY A FAX ?
YES
SCHEDULE "RECEIVE FAX"
15
NO
14
HAS A "COMPLETE" FAX BEEN RECEIVED ?
YES
SCHEDULE "ROUTE FAX"
17
IS THERE AN UNLOCKED FILE IN A DIRECTORY FOR INCOMING FAX MESSAGES
NO
16
IS NEW INCOMING ACTIVITY SOMETHING ELSE OF INTEREST ?
YES
SCHEDULE AN APPROPTIATE PROCESS
19
NO
18
INITIATE OUTGOING ACTIVITY ?
YES
SCHEDULE AN APPROPRIATE PROCESS
21
NO
20

FIG 2

Printed: 11-11-2004 ODOCP 02019563

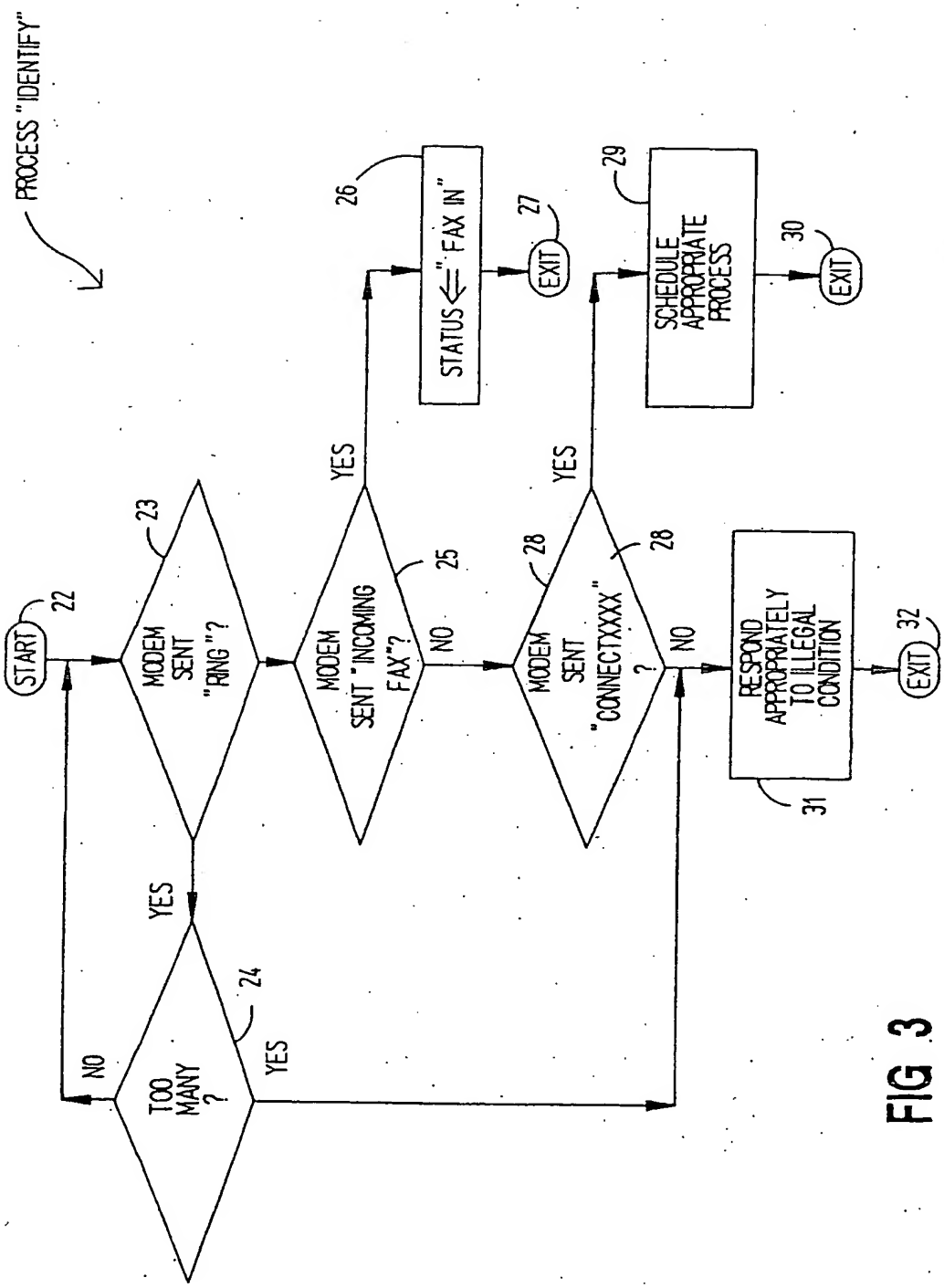

FIG 3

17 06-11-2004

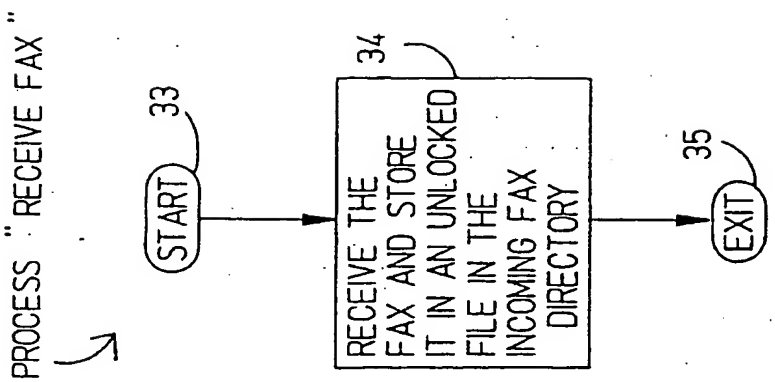
PROCESS "RECEIVE FAX"
START
33
RECEIVE THE FAX AND STORE IT IN AN UNLOCKED FILE IN THE INCOMING FAX DIRECTORY
34
EXIT
35

FIG 4

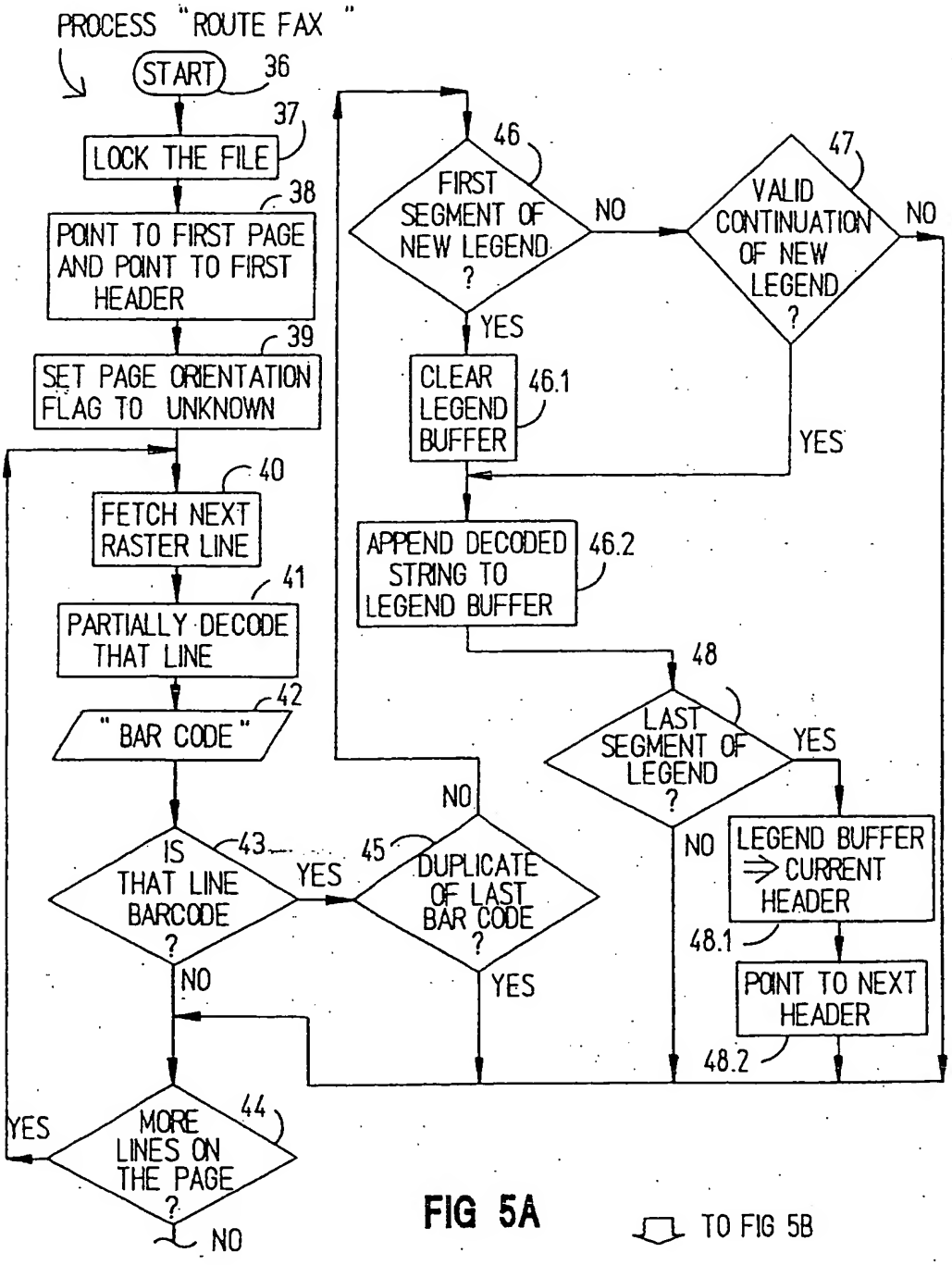
PROCESS "ROUTE FAX"
START
36
37
LOCK THE FILE
38
POINT TO FIRST PAGE AND POINT TO FIRST HEADER
39
SET PAGE ORIENTATION FLAG TO UNKNOWN
40
FETCH NEXT RASTER LINE
41
PARTIALLY DECODE THAT LINE
42
"BAR CODE"
43
IS THAT LINE BARCODE ?
YES
NO
45
DUPLICATE OF LAST BAR CODE ?
YES
NO
44
MORE LINES ON THE PAGE ?
YES
NO
46
FIRST SEGMENT OF NEW LEGEND ?
NO
YES
47
VALID CONTINUATION OF NEW LEGEND ?
NO
YES
46.1
CLEAR LEGEND BUFFER
46.2
APPEND DECODED STRING TO LEGEND BUFFER
48
LAST SEGMENT OF LEGEND ?
YES
NO
LEGEND BUFFER ⇒ CURRENT HEADER
48.1
POINT TO NEXT HEADER
48.2
TO FIG 5B

FIG 5A

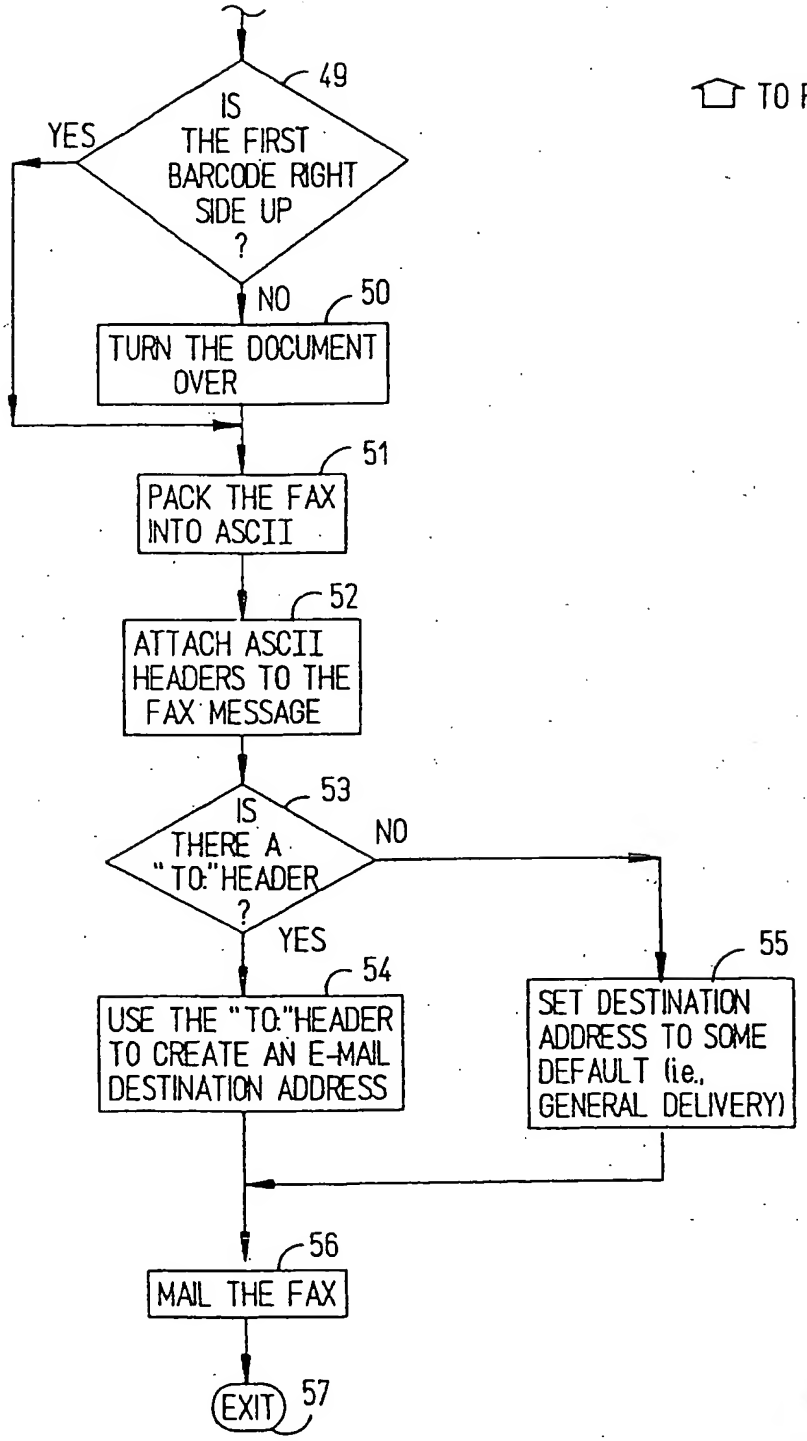
49
IS THE FIRST BARCODE RIGHT SIDE UP ?
YES
NO
50
TURN THE DOCUMENT OVER
51
PACK THE FAX INTO ASCII
52
ATTACH ASCII HEADERS TO THE FAX MESSAGE
53
IS THERE A "TO:" HEADER ?
NO
YES
54
USE THE "TO:" HEADER TO CREATE AN E-MAIL DESTINATION ADDRESS
55
SET DESTINATION ADDRESS TO SOME DEFAULT (i.e., GENERAL DELIVERY)
56
MAIL THE FAX
EXIT
57
TO FIG 5A

FIG 5B

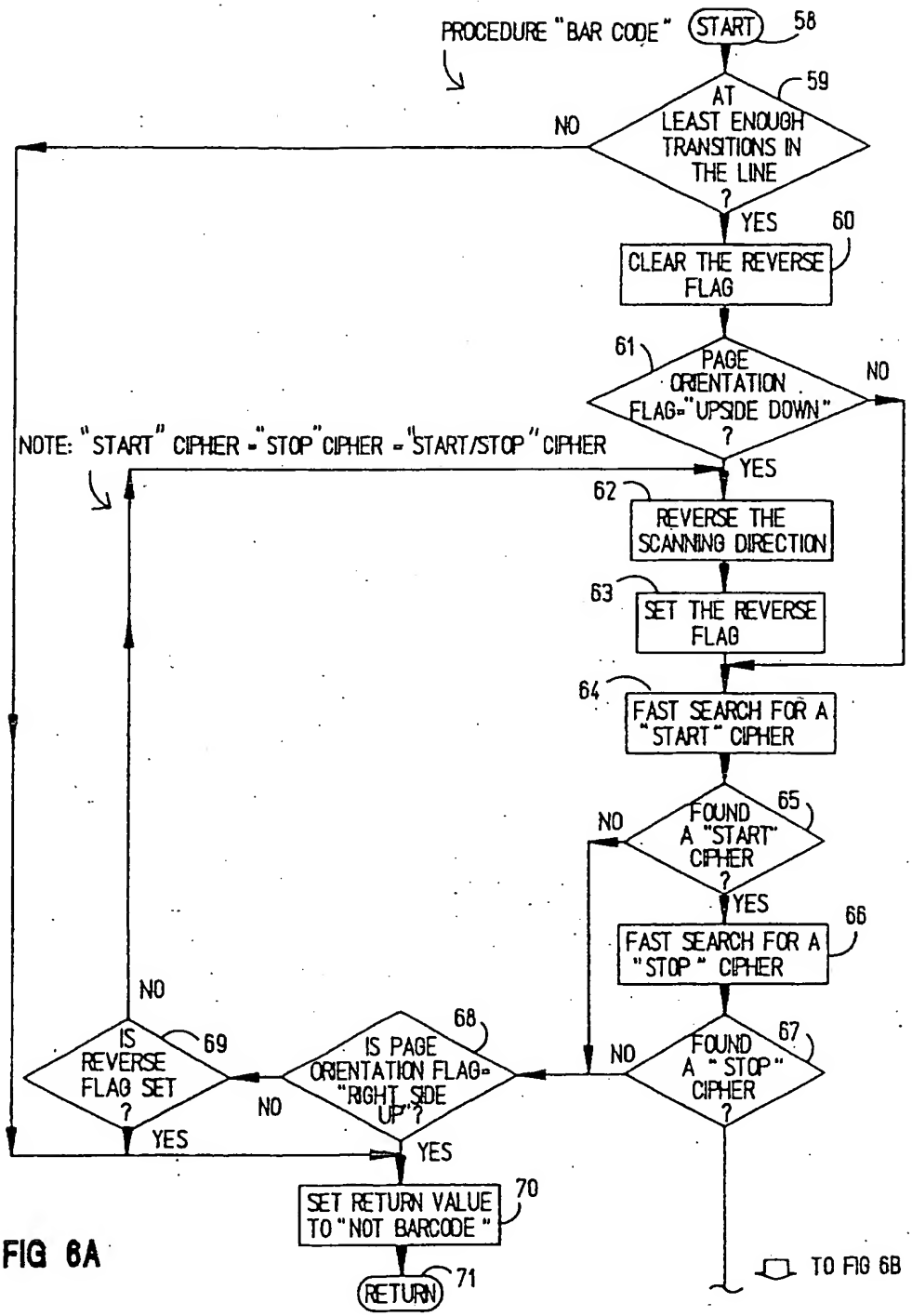
PROCEDURE "BAR CODE"
START
58
AT LEAST ENOUGH TRANSITIONS IN THE LINE ?
59
NO
YES
CLEAR THE REVERSE FLAG
60
PAGE ORIENTATION FLAG="UPSIDE DOWN" ?
61
NO
YES
NOTE: "START" CIPHER = "STOP" CIPHER = "START/STOP" CIPHER
REVERSE THE SCANNING DIRECTION
62
SET THE REVERSE FLAG
63
FAST SEARCH FOR A "START" CIPHER
64
FOUND A "START" CIPHER ?
65
NO
YES
FAST SEARCH FOR A "STOP" CIPHER
66
FOUND A "STOP" CIPHER ?
67
NO
IS PAGE ORIENTATION FLAG= "RIGHT SIDE UP"?
68
NO
YES
IS REVERSE FLAG SET ?
69
NO
YES
SET RETURN VALUE TO "NOT BARCODE"
70
RETURN
71
TO FIG 6B

FIG 6A

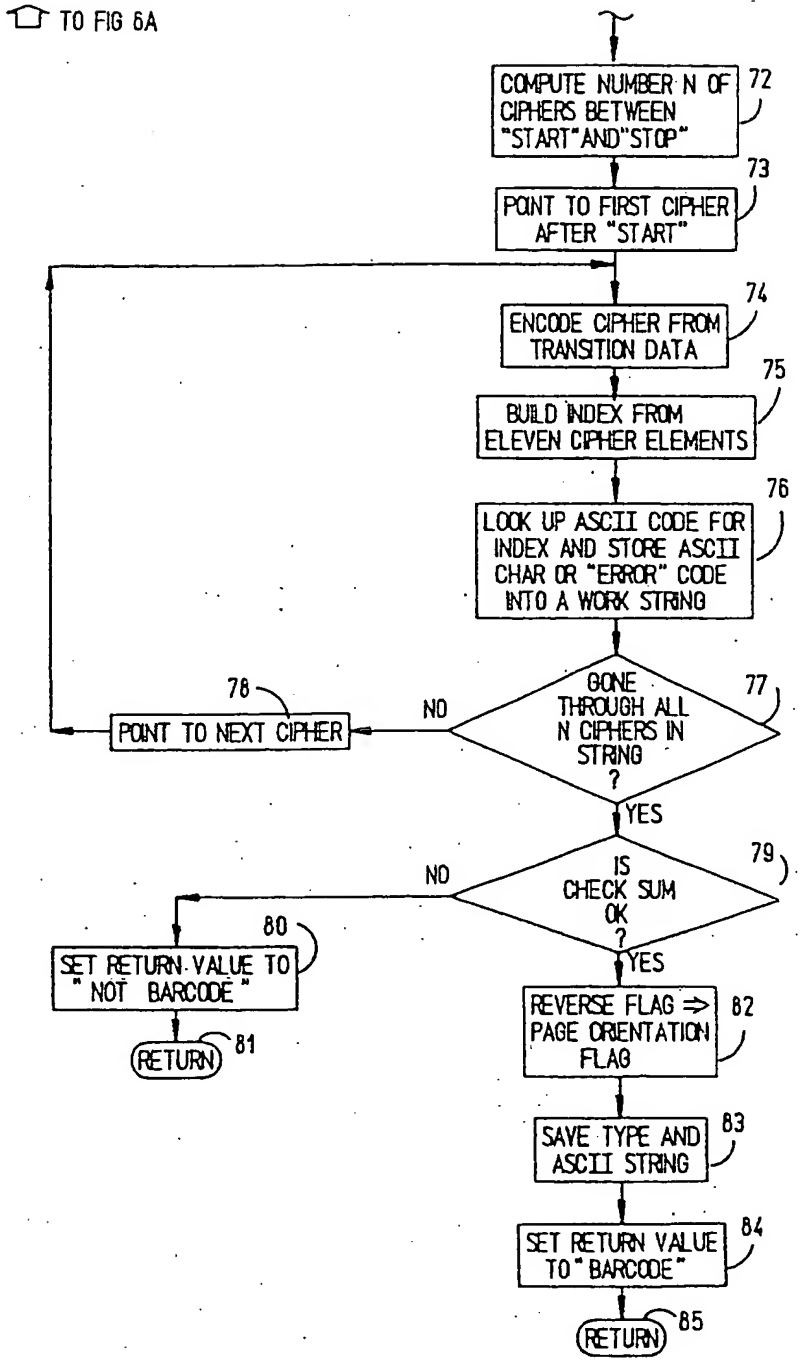
TO FIG 6A
COMPUTE NUMBER N OF CIPHERS BETWEEN "START" AND "STOP"
72
POINT TO FIRST CIPHER AFTER "START"
73
ENCODE CIPHER FROM TRANSITION DATA
74
BUILD INDEX FROM ELEVEN CIPHER ELEMENTS
75
LOOK UP ASCII CODE FOR INDEX AND STORE ASCII CHAR OR "ERROR" CODE INTO A WORK STRING
76
GONE THROUGH ALL N CIPHERS IN STRING ?
77
NO
POINT TO NEXT CIPHER
78
YES
IS CHECK SUM OK ?
79
NO
SET RETURN VALUE TO "NOT BARCODE"
80
RETURN
81
YES
REVERSE FLAG ⇒ PAGE ORIENTATION FLAG
82
SAVE TYPE AND ASCII STRING
83
SET RETURN VALUE TO "BARCODE"
84
RETURN
85

FIG 6B

Printed: 11-11-2004 EDOCP 02019563

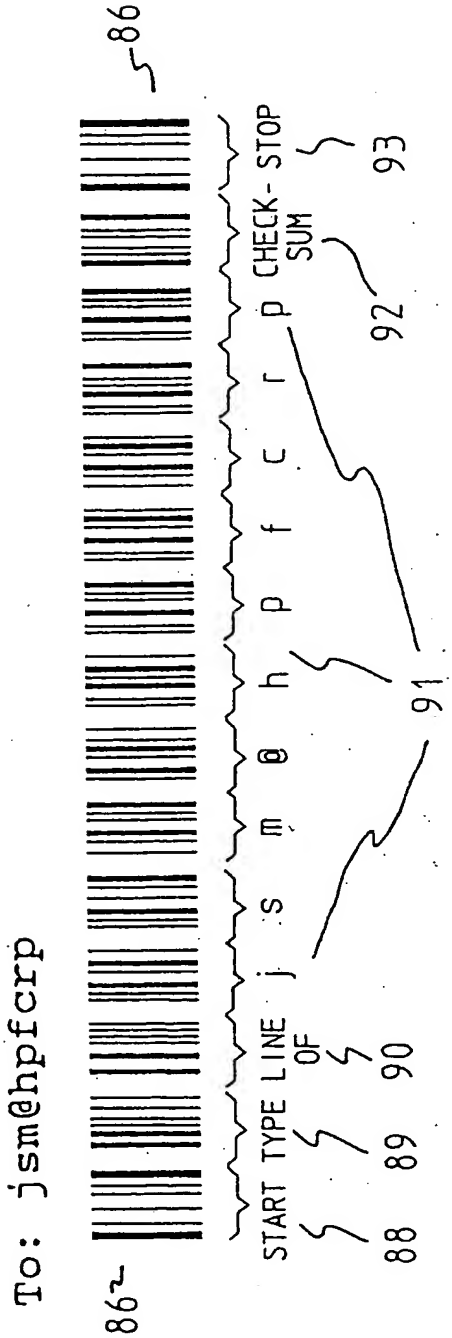

FIG 7

23 06-11-2004

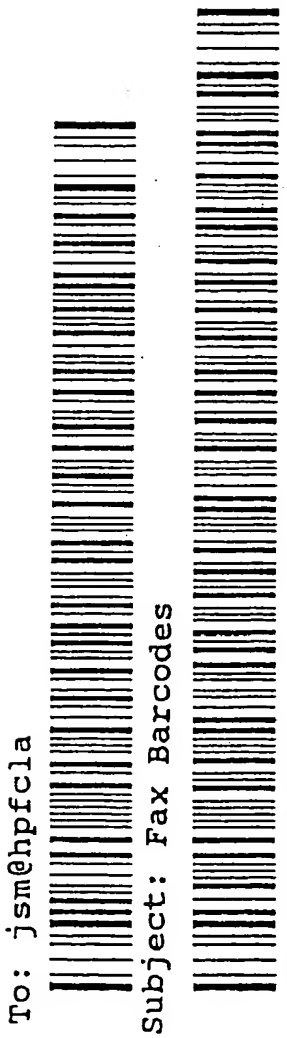

FIG 8A

Printed: 11-11-2004 EPODOCP 02019563

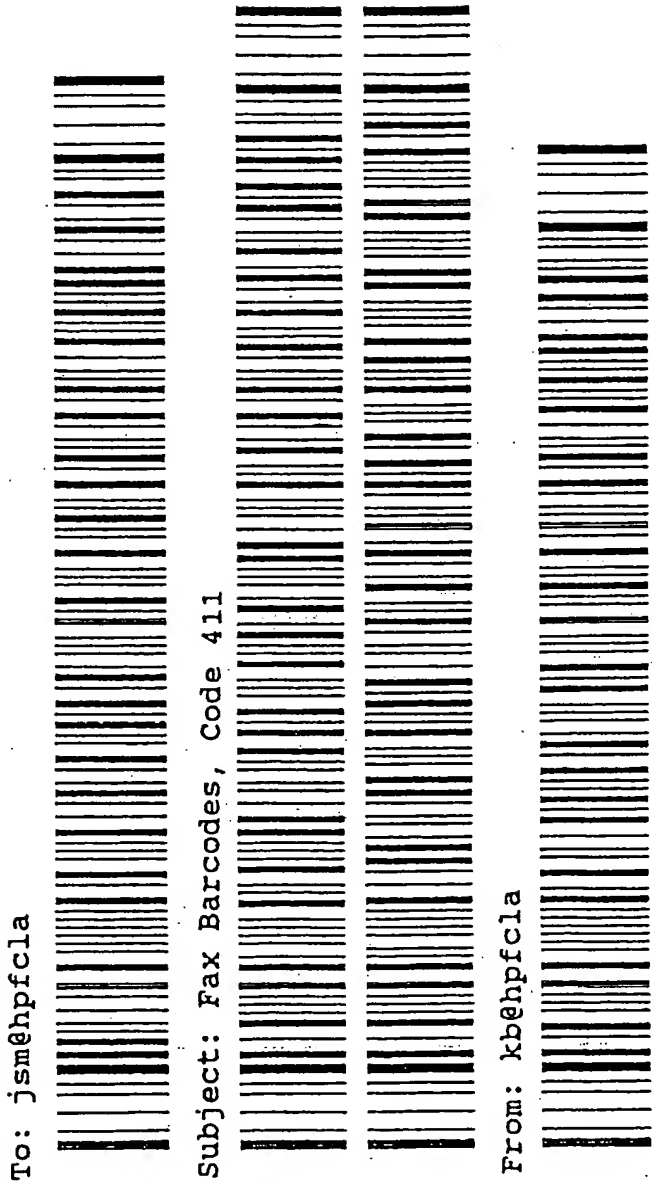

FIG 8B

25 06.11.2004

Printed: 11-11-2004 ODOCP 02019563

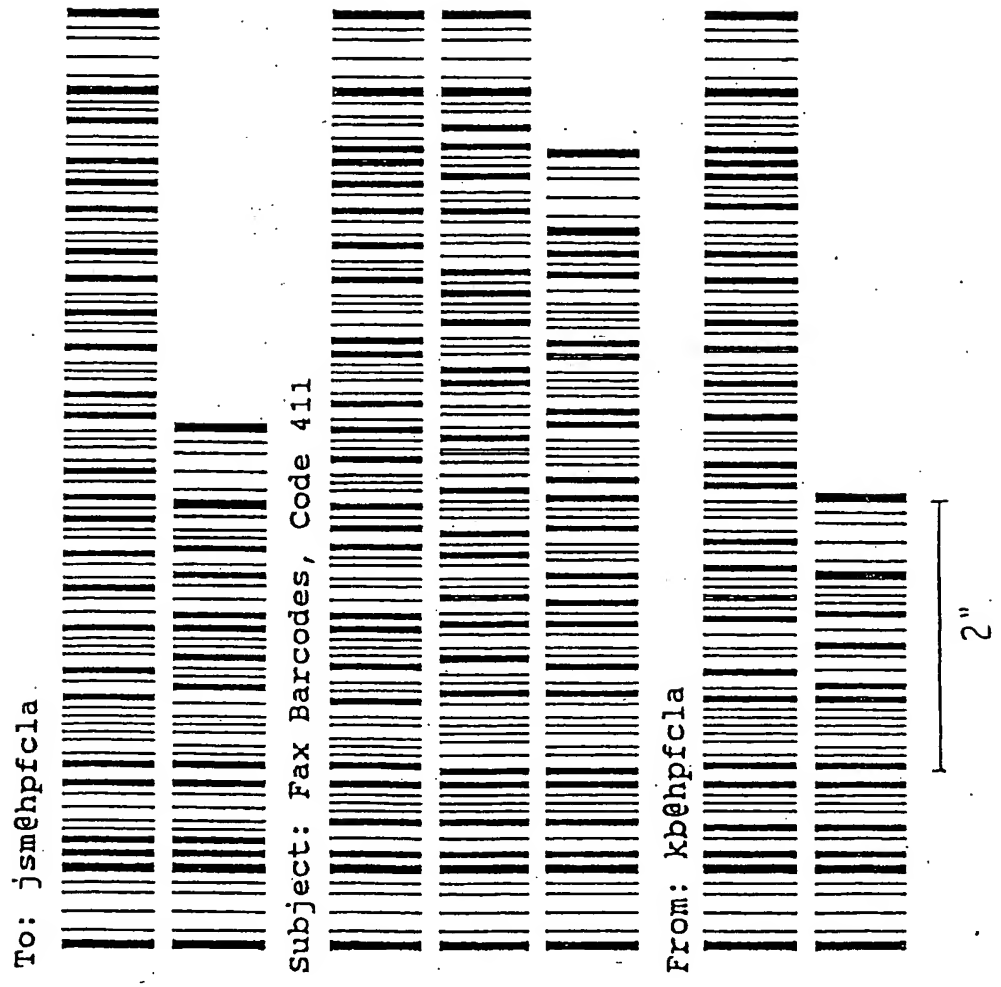

FIG 8C

26 06-11-2004

Printed: 11-11-2004 ODOCP 02019563

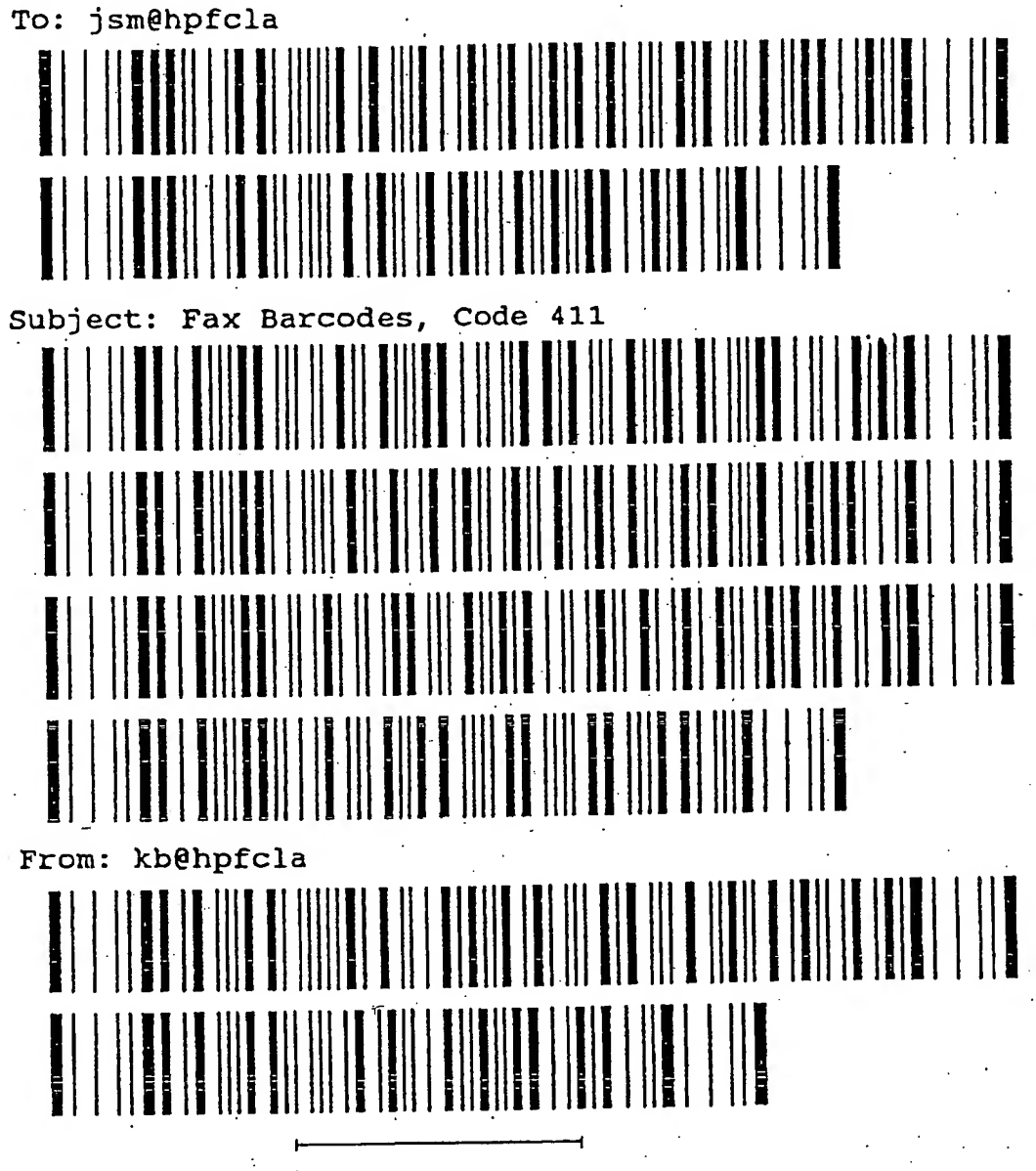

FIG 8D

27 06-11-2004

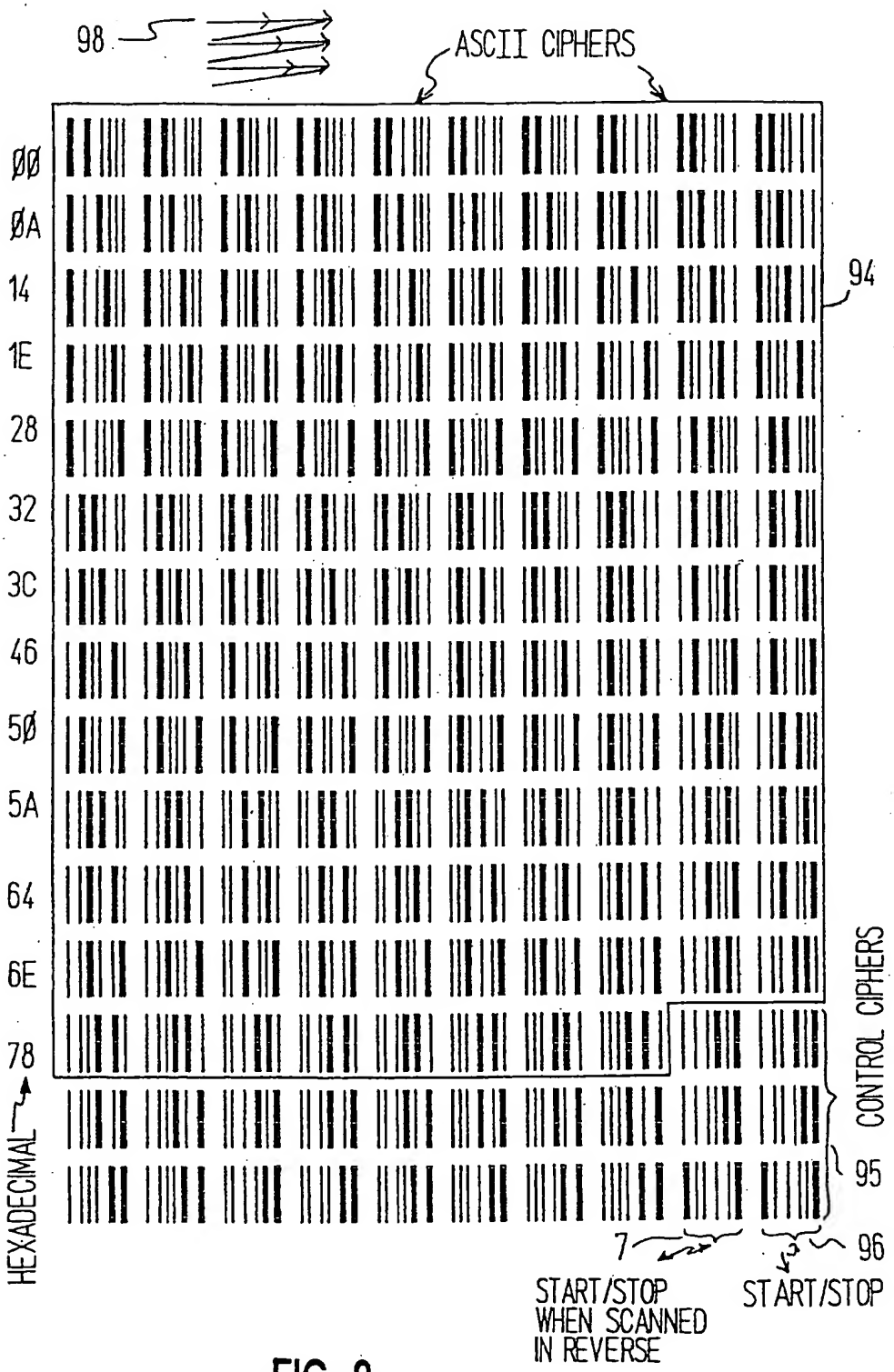
98
ASCII CIPHERS
ØØ
ØA
14
1E
28
32
3C
46
5Ø
5A
64
6E
78
HEXADECIMAL
94
CONTROL CIPHERS
95
96
7
START/STOP
WHEN SCANNED
IN REVERSE
START/STOP

FIG 9

Printed: 11-11-2004 EDOCP 02019563

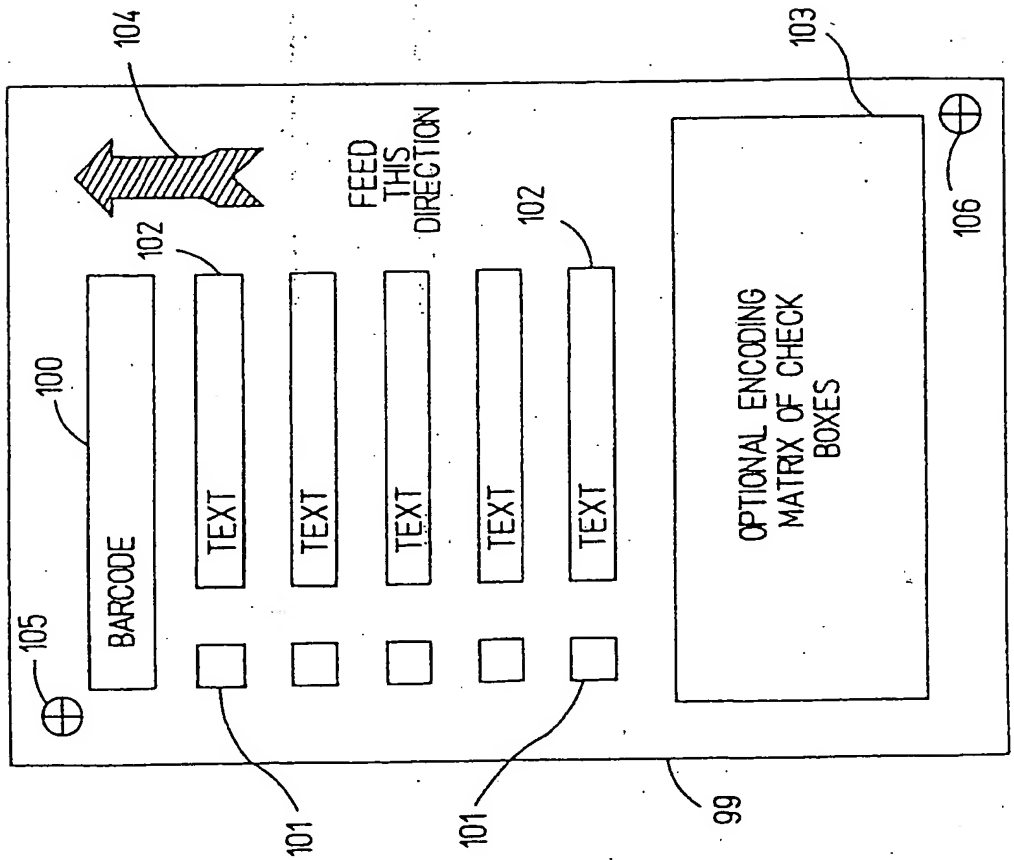

FIG 10

29 06-11-2004